Reciba notificaciones de la api de Servlet a través de Listeners

diego-chohfi
26/05/2021

En algunos sistemas web, debemos impedir que el usuario inicie sesión en más de una computadora. Pero la pregunta es, ¿dónde colocaríamos esta regla?

Se puede hacer justo cuando el usuario inicia sesión, para saber si ya hay un usuario con ese nombre de usuario, con una sesión abierta. Por supuesto, puedes organizarte y tener una sola clase que concentre la responsabilidad de trabajar con la sesión del usuario. No siempre tenemos la suerte de trabajar con un código más organizado, o puedes trabajar con un framework que esconde mucha cosa. Además, puedes estar interesado en ver los cambios en la sesión del usuario por una variedad de razones, no solo en el caso del inicio de sesión.

La especificación de Servlet de Java nos proporciona una función interesante, los **listeners**, donde podemos registrar varios **oyentes** por ciertos **eventos**. Siempre que ocurran estos eventos, en cualquier lugar del sistema, **seremos notificados**.

Para eso, la api tiene varias interfaces para registrar nuevos listeners y una de ellas es la HttpSessionAttributeListener, donde podemos saber cuándo un atributo fue *agregado, eliminado o reemplazado* de la sesión del usuario.

Estos métodos reciben como parámetro un objeto tipo HttpSessionBindingEvent y con él podemos recuperar las informaciones del atributo de la sesión, como el nombre y el objeto

pasado en el parámetro al [setAttribute] (http://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.6/api/javax/servlet/http/HttpSession.html#setAttribute(java.lang.String, java.lang.Object)) de la ya conocida clase **HttpSession**. Aquí está el esqueleto de la implementación:

```
@WebListener

public class UsuarioDuplicadoListener implements HttpSessionAttributeListener

    public void attributeAdded( HttpSessionBindingEvent bind ) {

    }

    public void attributeRemoved( HttpSessionBindingEvent bind ) {

    }

    public void attributeReplaced( HttpSessionBindingEvent arg0 ) {

    }

}
```

Con servlets 3.0, es suficiente la anotación @WebListener para que se cargue su listener. Para versiones anteriores, hay tags correspondientes para el web.xml.

Otros listeners también están disponibles para otros tipos de notificaciones, por ejemplo, si fuera necesario saber cuándo el proyecto se ejecutó por completo, el servidor podría notificarnos desde la interfaz ServletContextListener y para registrarlo, simplemente agregue la misma anotación que se agregó al listener anterior.

Pero, para implementar la funcionalidad que queremos, para evitar que el mismo usuario inicie sesión en más de un lugar, necesitamos saber si ese objeto agregado, para ese atributo, no fue agregado posteriormente por otra solicitud. Para eso tendríamos que acceder a todas las sesiones activas y, por motivos de seguridad, la especificación no lo permite.

Una alternativa es almacenar todos los usuarios conectados en una lista y, cuando otro usuario intenta iniciar sesión nuevamente, validar si el mismo objeto aún no se haya agregado.

Solo debemos tener cuidado con el tipo de lista en la que almacenaremos estas informaciones, es decir, como no queremos repeticiones podemos usar la interfaz *Set*, que no acepta elementos repetidos. Otra precaución que debemos tomar es que el ciclo de vida de un listener es el mismo que el de un Servlet, es decir, solo tendremos una instancia de `UsuarioDuplicadoListener` en la memoria, siendo compartido entre varios **threads**. Necesitamos una implementación de interfaz *Set* que sea **thread-safe** y la api de Java tiene algunas opciones que ya se ocupan de posibles problemas de competencia, o usan el `Collections.synchronizedSet`.

Supongamos por estándar que el nombre del atributo que gestiona el usuario conectado en el sistema es *usuarioConectado*, basta entonces que la clase que representa esta información implemente los métodos `es equals` y `hashCode` para que la colección maneje la búsqueda correctamente.

```
@WebListener

public class UsuarioDuplicadoListener implements HttpSessionAttributeListener
    private static final String ATTRIBUTE_NAME = "usuarioLogado";

    private static final Set<Object> USUARIOS_LOGADOS =
        Collections.synchronizedSet(new HashSet<Object>());

    public void attributeAdded(HttpSessionBindingEvent bind) {
        String attributeName = bind.getName();
        Object attributeValue = bind.getValue();
        if(ATTRIBUTE_NAME.equals(attributeName) ) {
            if(!USUARIOS_LOGADOS.add(attributeValue)){        bind.getSession(
            }
        }
    }

    public void attributeRemoved(HttpSessionBindingEvent bind) {
        String attributeName = bind.getName();
```

```
            Object attributeValue = bind.getValue();
            if(ATTRIBUTE_NAME.equals(attributeName)) {
                USUARIOS_LOGADOS.remove(attributeValue);
            }
        }

        public void attributeReplaced(HttpSessionBindingEvent bind) {
        }
    }
```

Con este enfoque, tenemos un código centralizado, independiente de la implementación que representa el usuario conectado en el sistema, y sería simple implementar alguna regla más específica: cómo bloquear al usuario anterior o incluso saber cuántos usuarios están conectados actualmente en el sistema. Varios otros problemas se podrían solucionar con **listeners** y la API de Java proporciona otras seis interfaces para escuchar varios eventos que pueden ocurrir en una aplicación web.

HttpSessionAttributeListener HttpSessionListener ServletRequestAttributeListener ServletRequestListener ServletContextAttributeListener

Este enfoque, de registrar un listener para validar si hay más de un usuario conectado en el sistema, es utilizado por **Spring Security.**

Puedes leer también:

- Conozca la API de fechas de Java 8
- Revisando la Orientación a Objetos: encapsulación de Java
- Cómo convertir de String para Date en Java

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento